**DAY 5**

Color detection by mouse click

Blob detection

Blob detection by mouse click

Color detection by mouse click

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

#include <math.h>

using namespace std;

using namespace cv;

int q=0, w=0;

void Call(int event, int x, int y, int flags, void\* userdata){

if (event ==EVENT\_LBUTTONDOWN){

q = x;

w = y;

}

}

int main(int argc, \_TCHAR\* argv[])

{

int i, a = 0, j, b = 0, c = 0, a1 = 255, b1 = 255, c1 = 255;

VideoCapture cap(0);

namedWindow("Contour", 1);

while (1){

setMouseCallback("Contour",Call,NULL);

Mat img;

cap >> img;

Mat img2 = img.clone();

Mat img1;

cvtColor(img, img1, CV\_BGR2HLS);

vector<Mat> channel;

split(img1, channel);

if (q != 0 && w != 0){

j = q;

i = w;

a = channel[0].at<uchar>(i, j) - 25;

a1 = channel[0].at<uchar>(i, j) + 25;

b = channel[1].at<uchar>(i, j) - 25;

b1 = channel[1].at<uchar>(i, j) + 25;

c = channel[2].at<uchar>(i, j) - 25;

c1 = channel[2].at<uchar>(i, j) + 25;

if (a <= 0) a = 0;

if (b <= 0) b = 0;

if (c <= 0) c = 0;

if (a1 >= 255) a1 = 255;

if (b1 >= 255) b1 = 255;

if (c1 >= 255) c1 = 255;

}

for (i = 0; i < img.rows; i++){

for (j = 0; j < img.cols; j++){

if (channel[0].at<uchar>(i, j)<a1 && channel[0].at<uchar>(i, j)>a) {

if (channel[1].at<uchar>(i, j)<b1 && channel[1].at<uchar>(i, j)>b) {

if (channel[2].at<uchar>(i, j)<c1 && channel[2].at<uchar>(i, j)>c) {

img.at<Vec3b>(i, j) = { 255, 255, 255 };

}

else img.at<Vec3b>(i, j) = { 0, 0, 0 };

}

else img.at<Vec3b>(i, j) = { 0, 0, 0 };

}

else img.at<Vec3b>(i, j) = { 0, 0, 0 };

}

}

Mat imga = img.clone();

GaussianBlur(img, img, Size(3, 3), 0, 0);

Canny(img, img, 50, 150,3);

GaussianBlur(img, img, Size(3, 3), 0, 0);

vector<vector<Point>> contours;

vector<Vec4i> hierarchy;

findContours(img, contours, hierarchy, CV\_RETR\_TREE, CV\_CHAIN\_APPROX\_SIMPLE, Point(0, 0));

int j,max=0;

for (i = 0; i < contours.size(); i++){

if (contours[i].size()>max) {

max = contours[i].size();

j = i;

}

}

drawContours(img2, contours, j, Scalar((a+a1)/2, (b+b1)/2, (c+c1)/2), 2, 8, hierarchy, 0, Point());

imshow("Contour", img2);

imshow("Canny", img);

imshow("Binary", imga);

if (waitKey(30) == 27) {

break;

}

}

return 0;

}

Shape Detection

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

#include <math.h>

using namespace std;

using namespace cv;

int main(int argc, \_TCHAR\* argv[])

{

int value = 100;

namedWindow("shape", 1);

createTrackbar("thresh", "shape", &value, 255);

VideoCapture cap(0);

while (1){

Mat img;

cap >> img;

cvtColor(img, img, CV\_RGB2GRAY);

Mat img1(img.rows, img.cols, CV\_8UC1, Scalar(0));

Mat img2 = img.clone();

vector<vector<Point>> contours;

vector<Vec4i> hierarchy;

int i, j;

for (i = 0; i < img.rows; i++){

for (j = 0; j < img.cols; j++){

if (img.at<uchar>(i, j)>value) img.at<uchar>(i, j) = 255;

else img.at<uchar>(i, j) = 0;

}

}

findContours(img, contours, hierarchy, CV\_RETR\_CCOMP, CV\_CHAIN\_APPROX\_NONE);

for (i = 0; i < contours.size(); i++){

vector<vector<Point>> result(contours.size());

approxPolyDP(contours[i], result[i], 3, true);

if (result[i].size() == 3){

line(img2, result[i][0], result[i][1],255, 4, 8, 0);

line(img2, result[i][1], result[i][2],255, 4, 8, 0);

line(img2, result[i][2], result[i][0],255, 4, 8, 0);

}

if (result[i].size() == 4){

line(img2, result[i][0], result[i][1], 255, 4, 8, 0);

line(img2, result[i][1], result[i][2], 255, 4, 8, 0);

line(img2, result[i][2], result[i][3], 255, 4, 8, 0);

line(img2, result[i][3], result[i][0], 255, 4, 8, 0);

}

if (result[i].size() > 4){

for(j = 0; j < result[i].size(); j++){

if (j == result[i].size() - 1) {

line(img2, result[i][j], result[i][0], 255, 4, 8, 0);

break;

}

line(img2, result[i][j], result[i][j+1], 255, 4, 8, 0);

}

}

}

imshow("shape", img2);

if (waitKey(30) == 27) break;

}

return 0;

}

Blob detection using bfs

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

#include <math.h>

#include <vector>

#include <deque>

using namespace std;

using namespace cv;

int main(int argc, \_TCHAR\* argv[])

{

int value = 150,i,j;

namedWindow("binary", 1);

createTrackbar("thresh", "binary", &value, 255);

while (1){

Mat img = imread("def.jpg", 1);

cvtColor(img, img, CV\_RGB2GRAY);

Mat img1(img.rows, img.cols, CV\_8UC1);

img1 = Scalar(0);

for (i = 0; i < img1.rows; i++){

for (j = 0; j < img1.cols; j++){

if (img.at<uchar>(i, j) >= value) img1.at<uchar>(i, j) = 0;

else img1.at<uchar>(i, j) = 255;

}

}

Mat imgo = img1.clone();

Mat img2(img.rows, img.cols, CV\_8UC3);

img2 = Scalar(255,255,255);

deque<Point> q;

int k, l, x[3] = { 150 }, y = 2 ,z=1, i1, j1;

Point pt;

for (i = 0; i < img.rows; i++){

for (j = 0; j < img.cols; j++){

if (img1.at<uchar>(i, j) == 255 ) {

q.push\_front(Point(j, i));

y--;

if (y < 0) y = 2;

z--;

if (z < 0) z = 2;

x[y] = x[y] - 37;

if (x[y] <= 0) x[y] = 230;

while (!q.empty()){

pt=q.back();

q.pop\_back();

i1 = pt.y;

j1 = pt.x;

for (k = i1 + 1; k > i1 - 2; k--){

for (l = j1 + 1; l > j1 - 2; l--){

if (img1.at<uchar>(k, l) == 255 ) {

q.push\_front(Point(l, k));

img2.at<Vec3b>(k, l)[y] = x[y];

img2.at<Vec3b>(k, l)[z] = 255-x[y];

img2.at<Vec3b>(k, l)[(y+z)/2] =( x[y]+255)/2;

img1.at<uchar>(k, l) = 0;

}

}

}

}

}

}

}

imshow("binary", img2);

imshow("gray", imgo);

if (waitKey(30) == 27) break;

}

return 0;

}

Blob detection by mouse click

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

#include <math.h>

#include <vector>

#include <deque>

using namespace std;

using namespace cv;

int q=0,w=0;

void Call(int event, int x, int y, int flags, void\* userdata){

if (event == EVENT\_LBUTTONDOWN){

q = x;

w = y;

}

}

int main(int argc, \_TCHAR\* argv[])

{

int value = 100;

int i,j,v1=0,v2=250;

namedWindow("binary", 1);

//createTrackbar("thresh", "binary", &value, 255);

while (1){

setMouseCallback("binary", Call, NULL);

Mat img = imread("def.jpg", 1);

Mat imgo = img.clone();

cvtColor(img, img, CV\_RGB2GRAY);

if (q != 0 && w != 0){

value = img.at<uchar>(w, q);

}

Mat img1(img.rows, img.cols, CV\_8UC1);

img1 = Scalar(0);

for (i = 0; i < img1.rows; i++){

for (j = 0; j < img1.cols; j++){

v1 = value - 5;

v2 = value + 5;

if (v1 < 0) v1 = 0;

if (v2 > 255) v2 = 255;

if (img.at<uchar>(i, j) > v1&&img.at<uchar>(i,j)<v2) img1.at<uchar>(i, j) = 255;

else img1.at<uchar>(i, j) = 0;

}

}

Mat img2(img.rows, img.cols, CV\_8UC3);

img2 = Scalar(255,255,255);

deque<Point> q;

int k, l, x[3] = { 175 }, y = 2,z=0, i1, j1;

Point pt;

for (i = 0; i < img.rows; i++){

for (j = 0; j < img.cols; j++){

if (img1.at<uchar>(i, j) == 255 ) {

q.push\_front(Point(j, i));

y--;

if (y < 0) y = 2;

z--;

if (z < 0) z = 2;

x[y] = x[y] - 37;

if (x[y] <= 0) x[y] = 230;

while (!q.empty()){

pt=q.back();

q.pop\_back();

i1 = pt.y;

j1 = pt.x;

for (k = i1 + 1; k > i1 - 2; k--){

for (l = j1 + 1; l > j1 - 2; l--){

if (img1.at<uchar>(k, l) == 255 ) {

q.push\_front(Point(l, k));

img2.at<Vec3b>(k, l)[y] = x[y];

img2.at<Vec3b>(k, l)[z] = 255-x[y];

img2.at<Vec3b>(k, l)[(y+z)/2] =( x[y]+255)/2;

img1.at<uchar>(k, l) = 0;

}

}

}

}

}

}

}

imshow("binary", imgo);

imshow("graph", img2);

if (waitKey(30) == 27) break;

}

return 0;

}